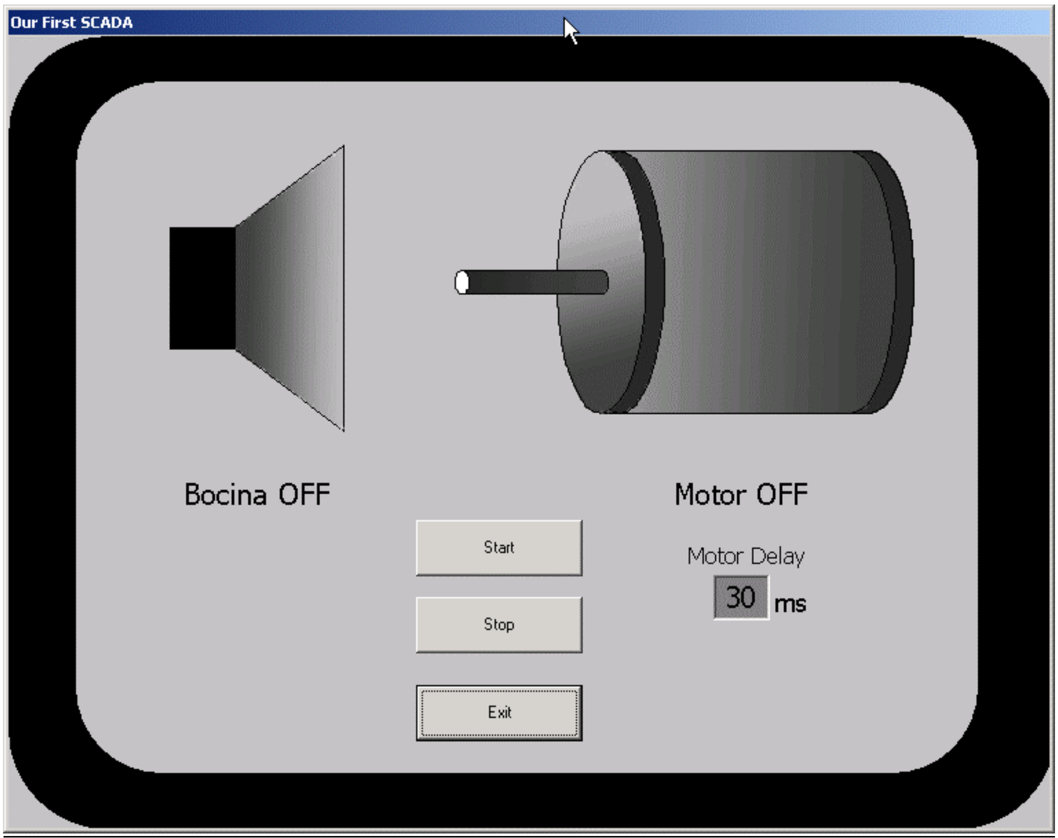
VB_ScadaLadder

¿Que es VB_ScadaLadder?

VB_ScadaLadder es el SCADA (Supervisory Control And Data Acquisition) para los MLCHIPs de Microladder (desarrollado para ML-CHIP1 y ML-CHIP2). VB_ScadaLadder está compuesto de dos librerías (DLL), llamadas ML_DLLVB_1_0 y ML_DLLVC_1_0. Gracias a estas dos librerías se pueden desarrollar programas en Visual Basic y comunicar con los MLCHIP1 o MLCHIP2, de manera que puede desarrollar su propio SCADA, controlar el rendimiento y funcionamiento de su MLCHIP y monitorizar sus variables.

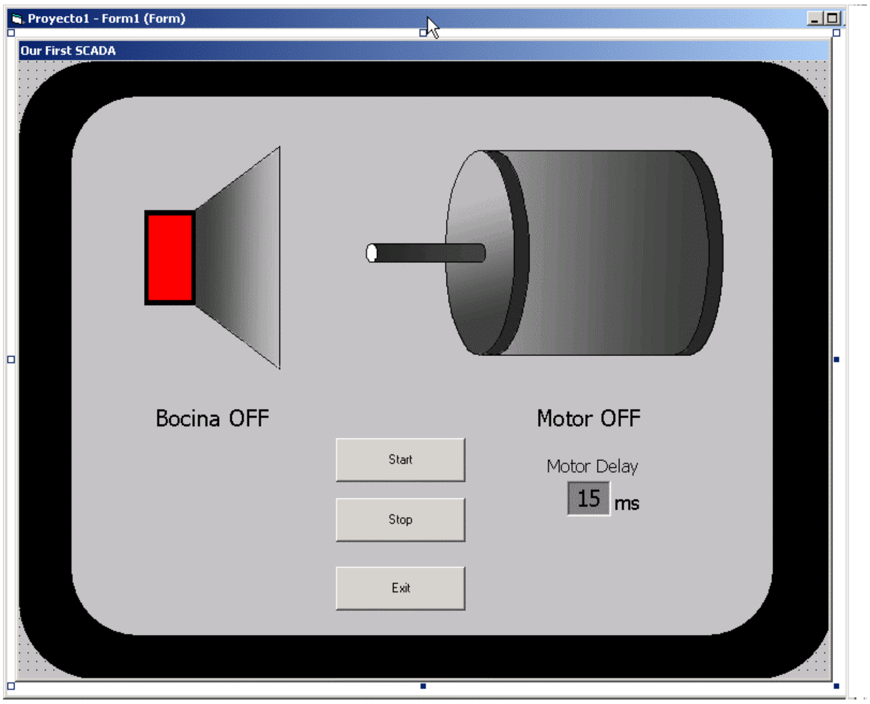
Empecemos con un ejemplo

Supongamos un sistema formado por una BOCINA y un MOTOR, el usuario puede encender el motor desde el PC pulsando un botón llamado "Start" y pararlo con otro botón llamado 'Stop'.

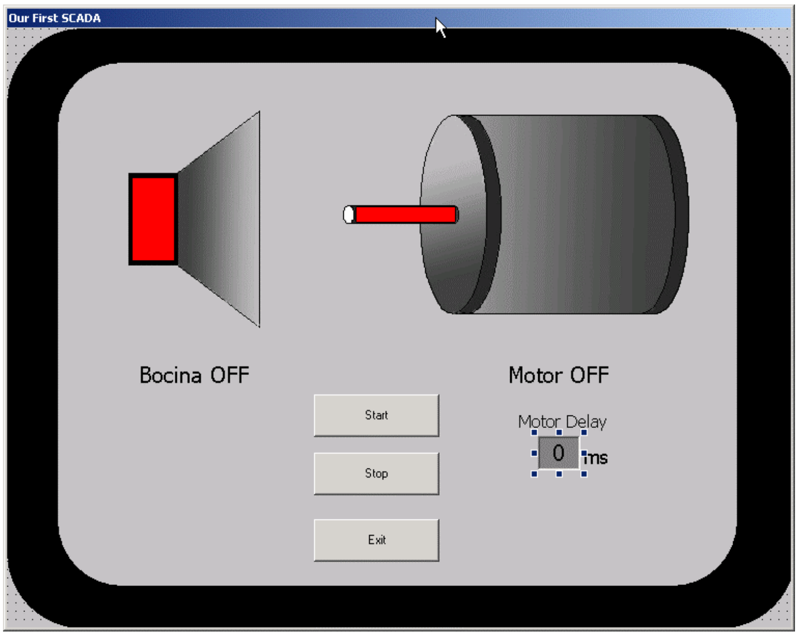


Ejemplo de control de un motor y bocina

Cuando el usuario pulsa el botón "Start" una bocina empieza a sonar, avisando del inminente encendido del motor tras un periodo de 3 segundos.



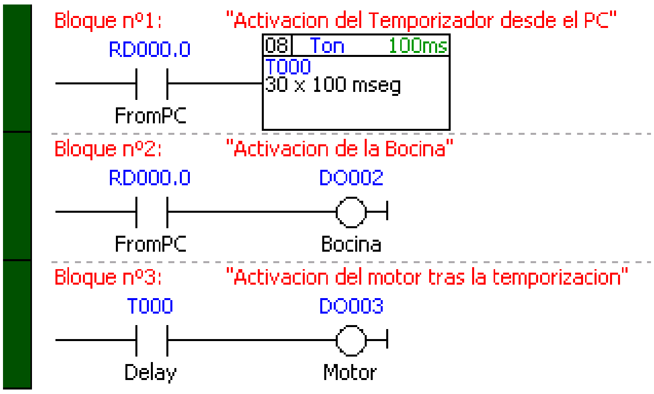
En la imagen se puede observar como se ilumina la imagen de la bocina a la vez que se visualiza el tiempo restante para que se encienda el motor.



Cuando el tiempo llega a "0" el motor arranca y la imagen del motor en el SCADA se enciende. Puede apagar el motor pulsando el botón Stop

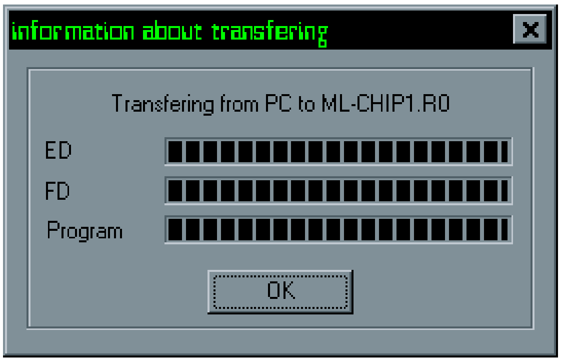
Como programar el ejemplo anterior.

Asumimos que tiene conocimientos básicos de programación en MICROLADDER y de programación en Visual Basic.
En primer lugar tenemos que programar el ML_CHIP1 para tener el control sobre la bocina y el motor.



Programa ladder

Tenemos un timer que se activará con una orden desde el PC enviada por el usuario al pulsar el botón 'Start'. Al pulsar este botón se envía al MLCHIP una orden para activar RD000.0 (El botón 'Stop' desactiva este mismo bit)
Como se puede apreciar en el bloque nº2, mientras que RD000.0 este activado la salida DO002 estará activa también, Cuando el timer (T000) expira (bloque nº3) la salida digital DO003 se activa y el motor arranca. Como T000 es un Timer tipo TON cuando RD000.0 pase a estar inactivo (al pulsar 'Stop') ambos bocina y motor pararán.

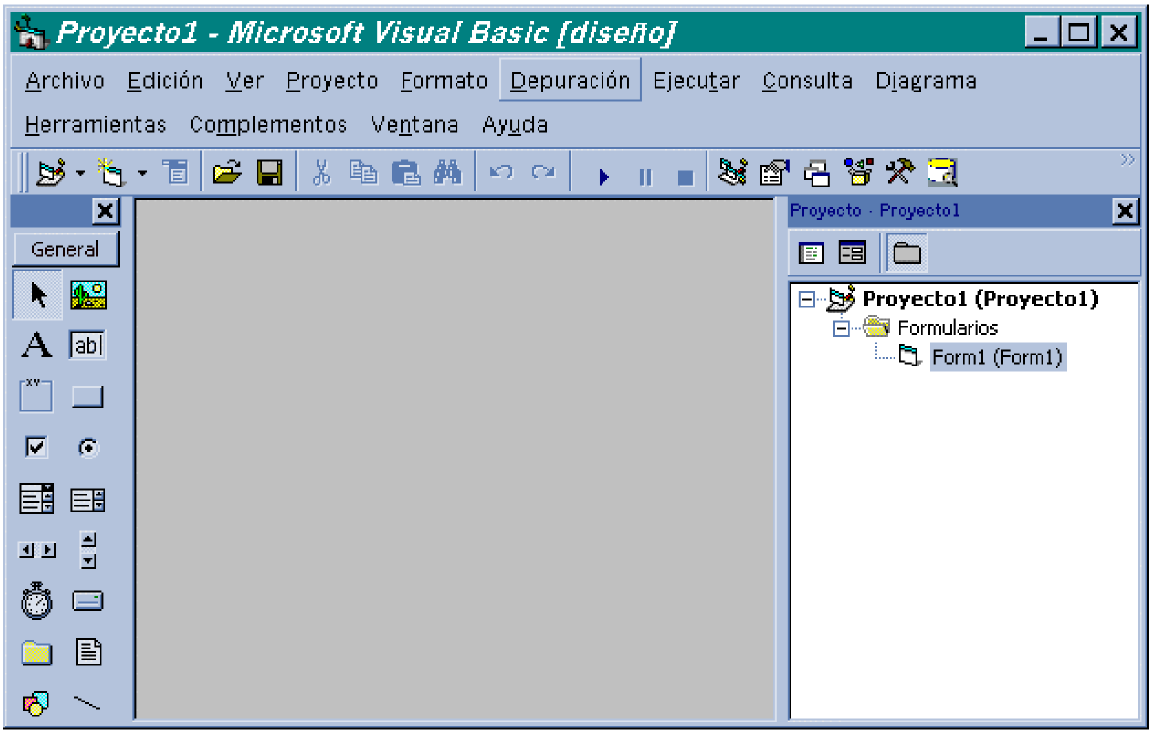


Transferencia del programa a ML-CHIP1

De esta manera ML-CHIP1 está preparado para controlar el sistema.

Ahora debemos programar en Visual Basic para realizar el SCADA.

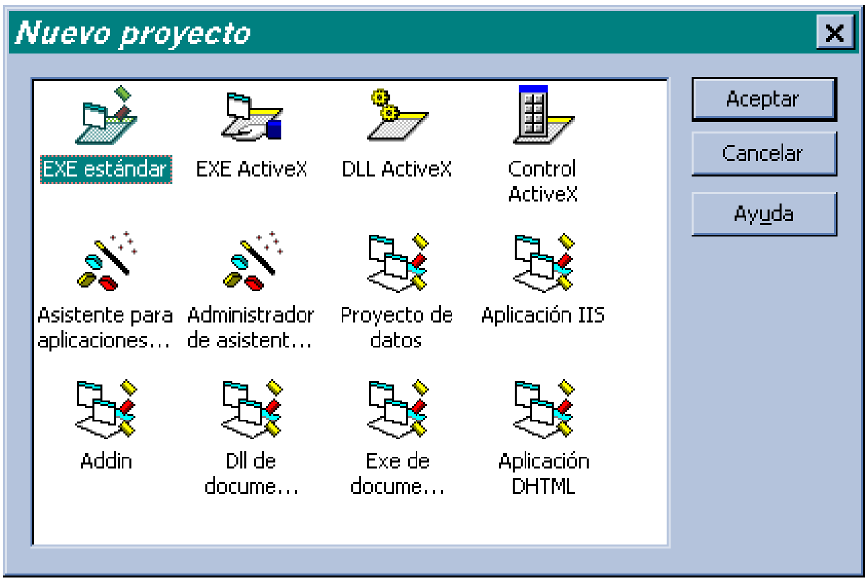
Lo primero es situar las DLL's de Microladder en el Directorio de Windows [c:\Windows] (o [c:\Winnt] en caso de ser Windows NT etc...) a continuación ya podemos abrir Visual Basic..



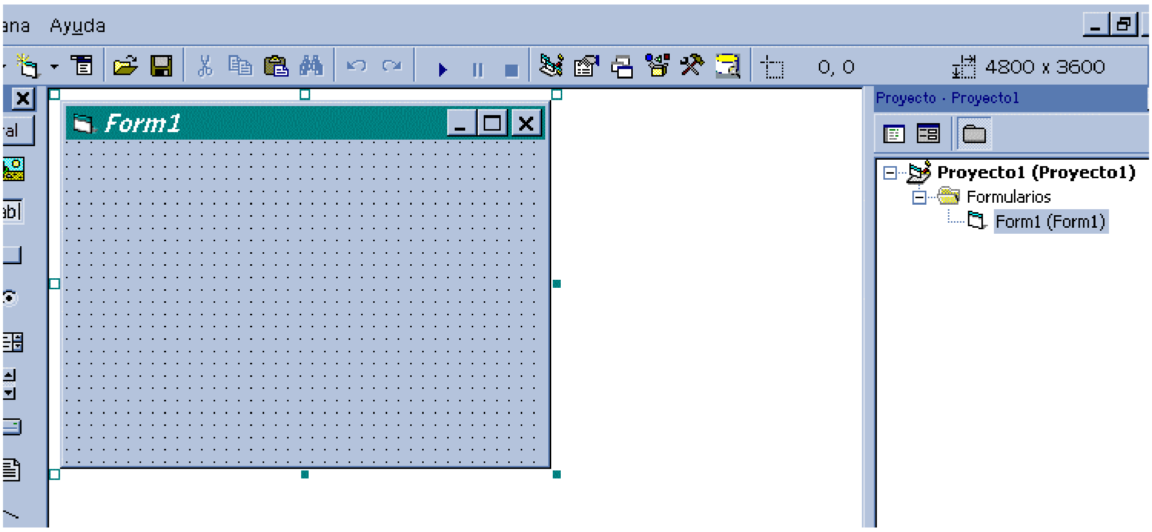
Entorno Visual Basic



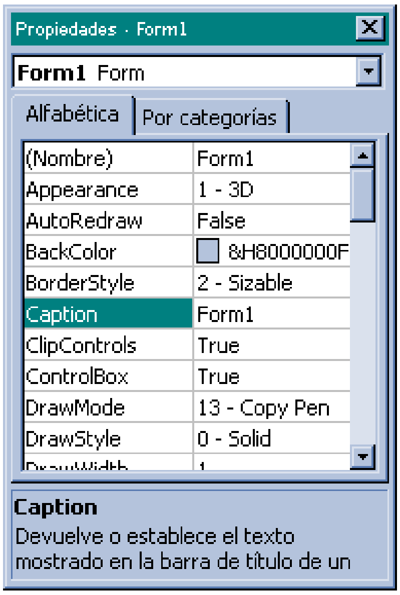
Crear Proyecto nuevo



Selección "EXE estándar"

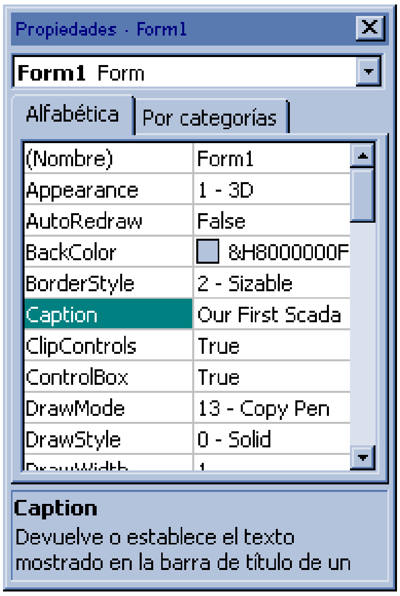


Presionando F4 abrimos la ventana de Propiedades

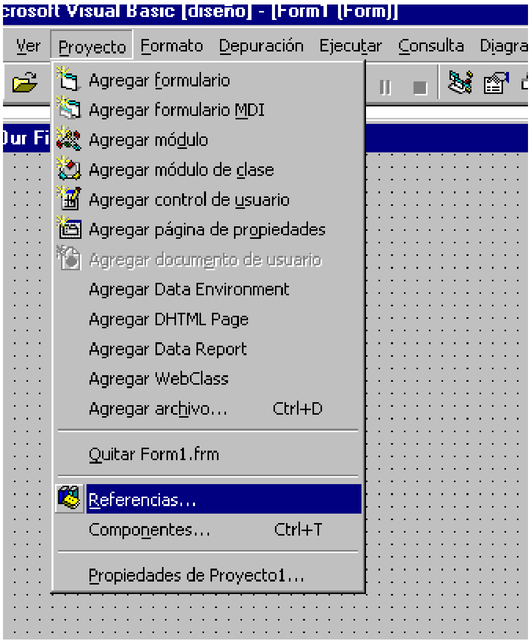


A partir de ahora, puede acceder al formulario y sus componentes a través del Nombre 'Form1'

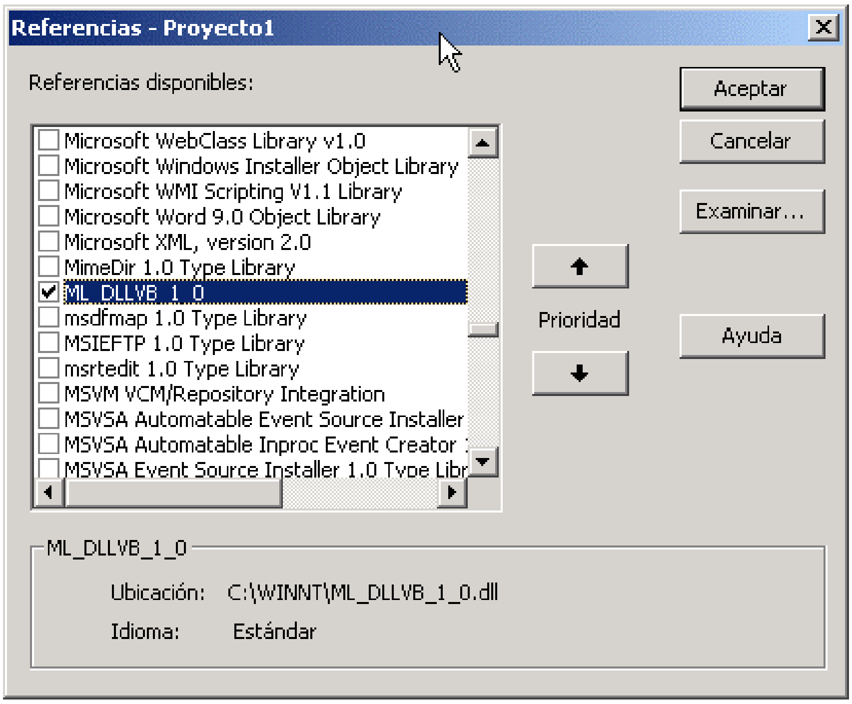
En esta ventana puede controlar las propiedades del formulario, (El formulario es la ventana principal de nuestro SCADA)
Puede definir el título de nuestro formulario modificando el campo 'Caption'. Vamos a llamarlo por ejemplo 'Nuestro Primer SCADA'



El siguiente paso es enlazar nuestro proyecto con las DDL de Microladder.

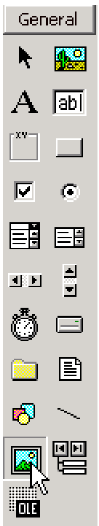


Opción para enlazar con las DDL de Microladder

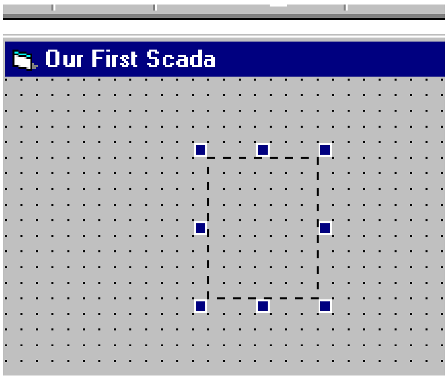


Enlace con la Librería ML_DLLVB_1_0

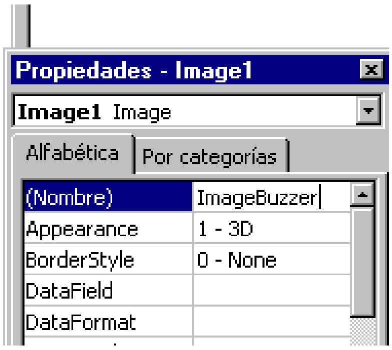
Como se puede apreciar solo debemos enlazar con esta librería. La Librería *ML_DLLVC_1_0.* basta con este ubicarla en el directorio previamente especificado, y no necesita ser enlazada desde Visual Basic.



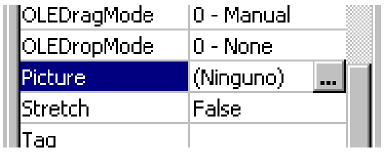
Insertar una imagen



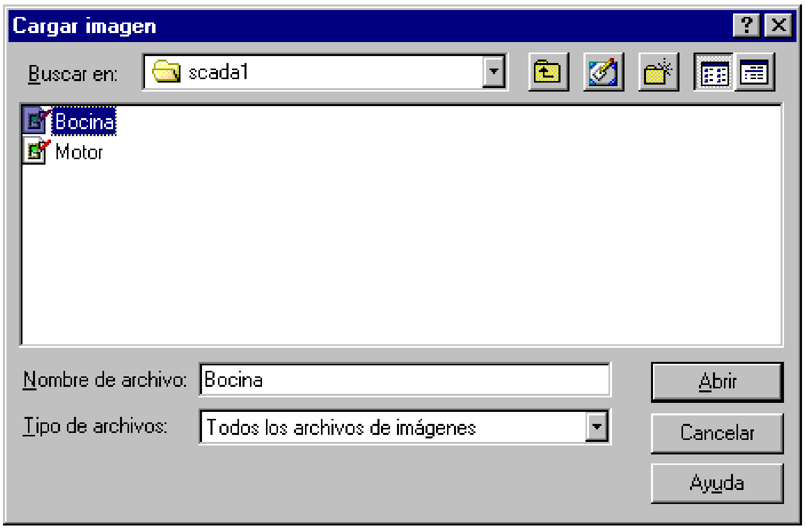
Selección del área sonde vamos a colocar la imagen



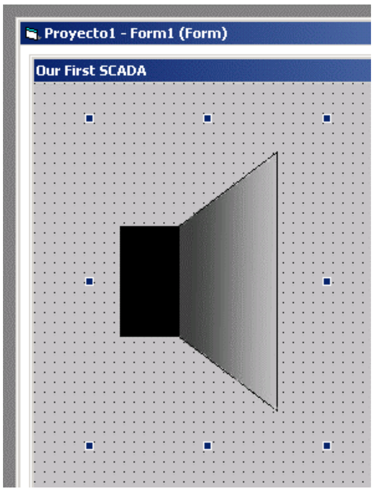
Renombramos la imagen



Enlazamos con la imagen de la bocina pulsando
en los puntos suspensivos

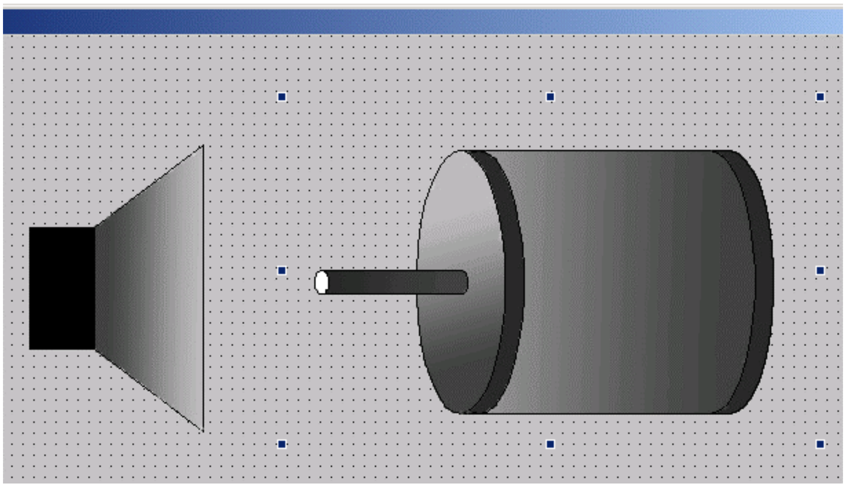


Selección de la imagen que va a representar la bocina



Inserción de la imagen seleccionada de la Bocina.

Realizamos los mismos pasos para insertar la imagen del motor

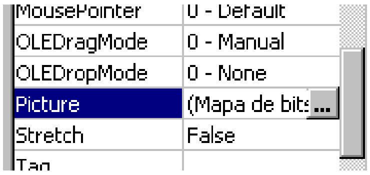


Inserción de la imagen del Motor.

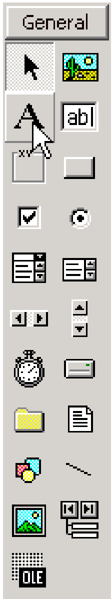
Asignamos a la propiedad (Nombre) de esta imagen: (Nombre) = ImageMotor.



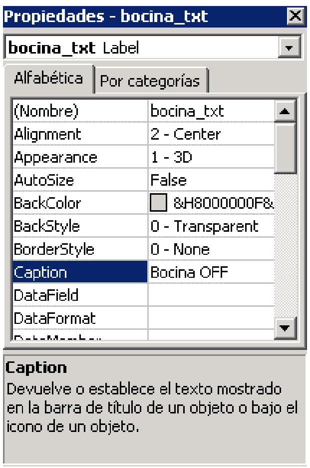
Le asociamos la imagen que consideremos adecuada



Ahora vamos a insertar en el formulario un Etiqueta (Label) para identificar la bocina, solo a efectos visuales, para el usuario del SCADA:

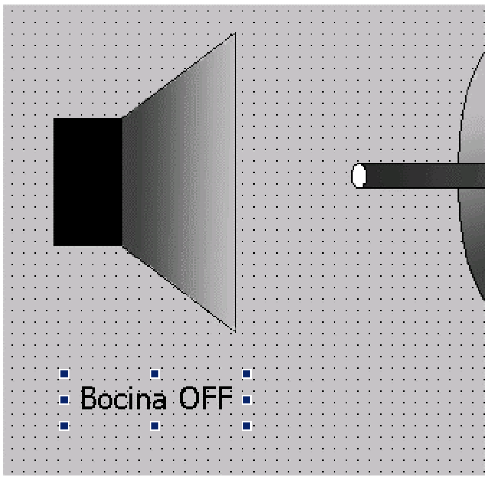


Insertar Label

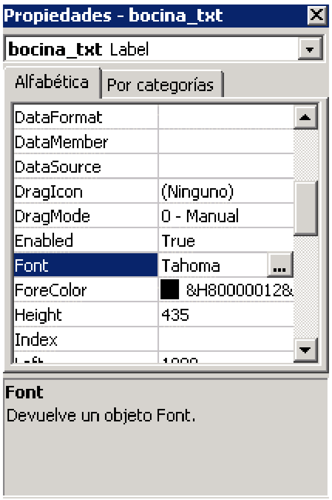


Cambiamos el Texto de 'Label1' por el de 'Bocina OFF'

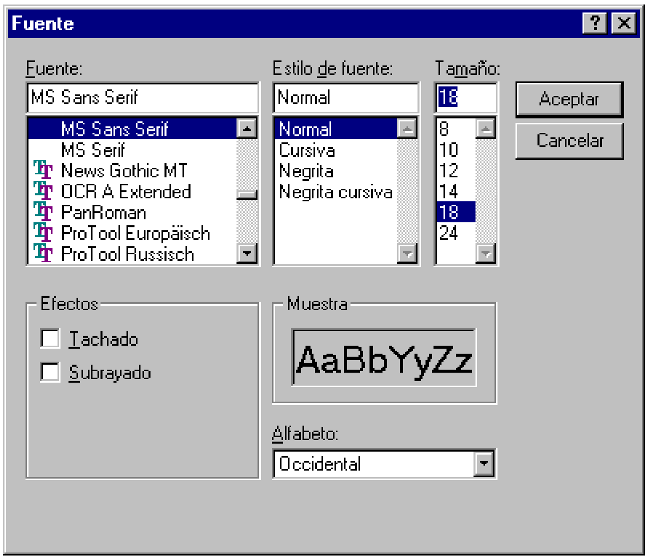
A partir de ahora, Esta etiqueta (label) será accesible por el nombre 'Bocina_txt', modificable en el campo '(Nombre)'. Podemos modificar la fuente de la etiqueta a través del campo 'Font'.



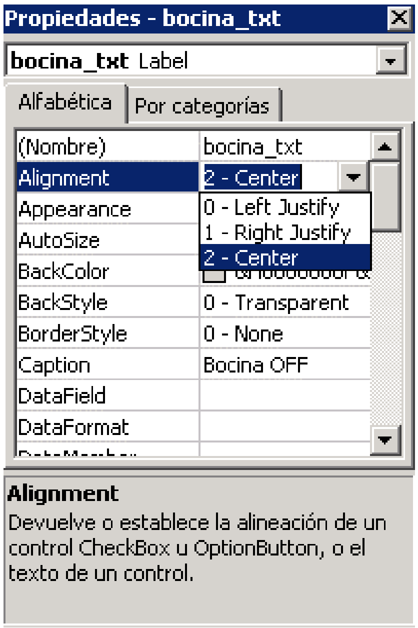
La situamos debajo de la imagen de la bocina



Haciendo clic sobre los puntos suspensivos aparece el siguiente dialogo:



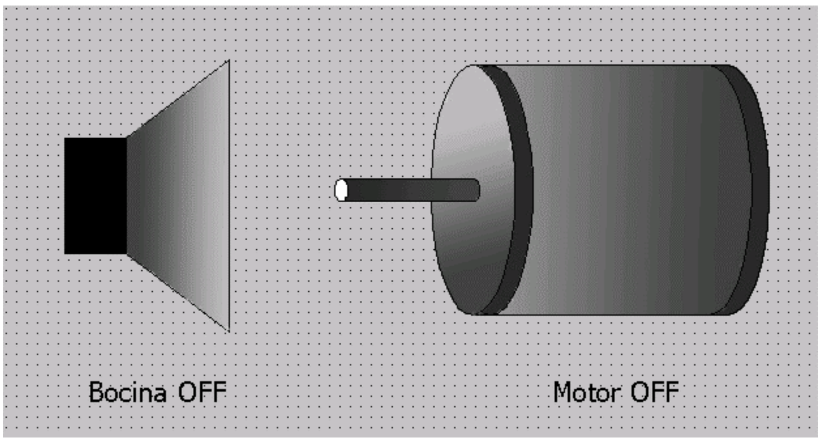
Nosotros hemos seleccionado como fuente ‘Tahoma’ y el tamaño de 16 ptos.



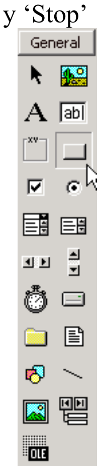
Cambiamos también la alineación del texto a 'Center'

De está manera conseguimos que el texto tenga el aspecto deseado

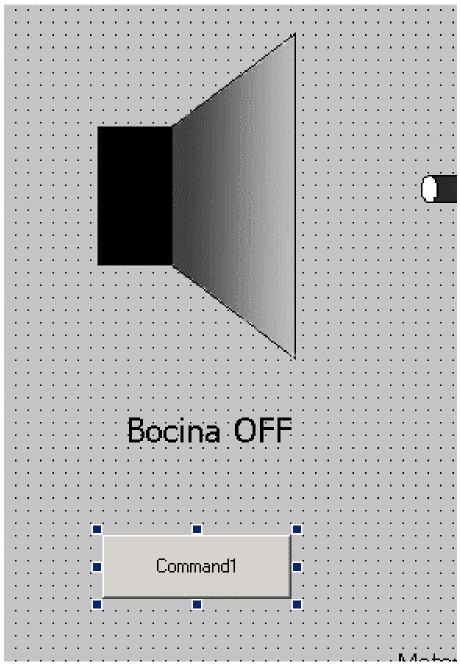
Hacemos lo mismo para ponerle una etiqueta al motor



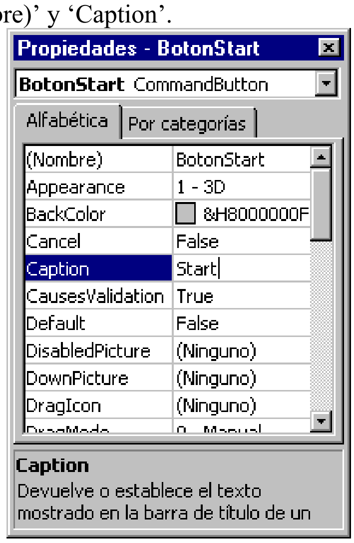
A continuación añadiremos los botones de 'Start' y 'Stop'



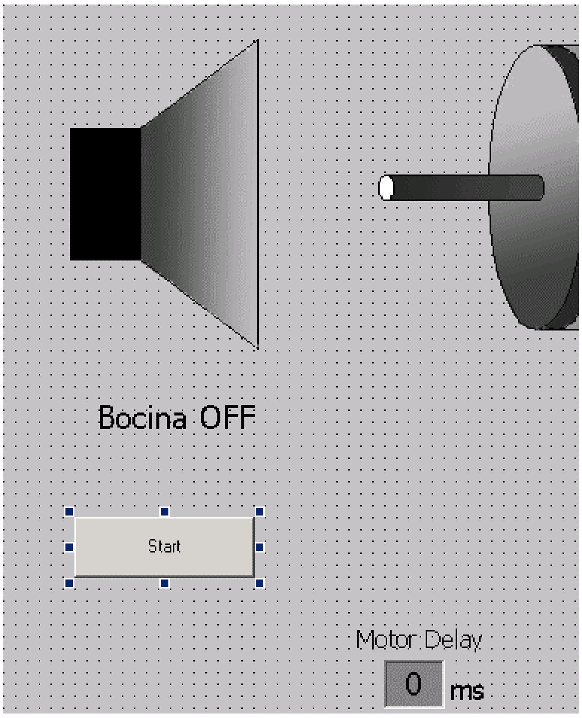
Insertar botón



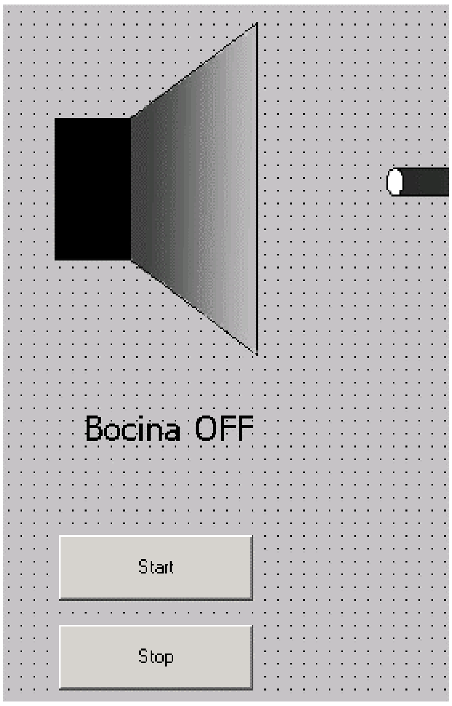
Modificamos sus propiedades '(Nombre)' y 'Caption'.



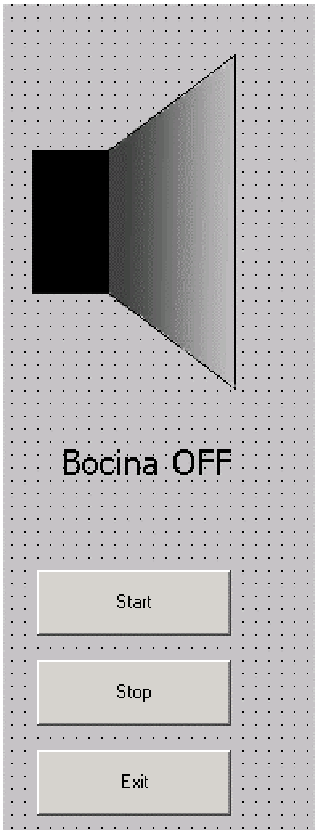
Así conseguimos:



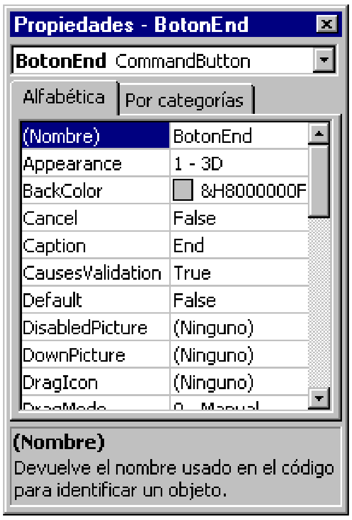
Añadimos manera similar el botón de 'Stop'



Y el botón de “Exit” (para finalizar el programa)



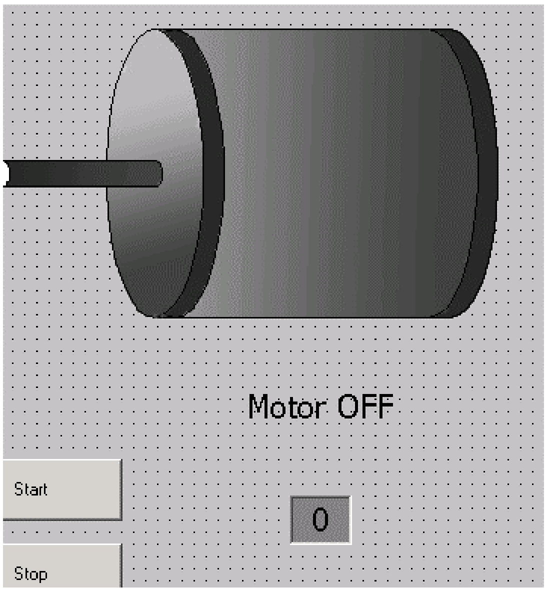
Le Asignamos las siguientes propiedades:



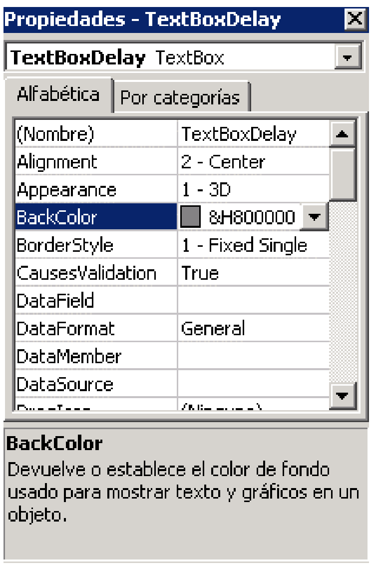
Añadimos la TextBox para mostrar el tiempo restante para la activación del motor, seleccionamos para ello la herramienta de ‘TextBox’.



Añadir Text Box



Inserción de Text Box

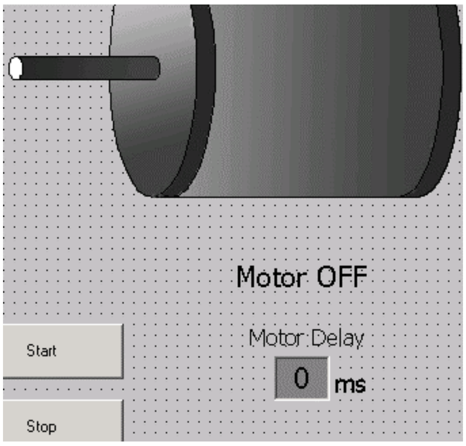


Y le fijamos los campos ‘BackColor’ con un gris más oscuro y el campo ‘Text’ con el valor “0”

Le añadimos también una par de etiquetas (labels) llamadas:

- Delay Motor.
- ms.

Y las situamos tal y como muestra la imagen siguiente:



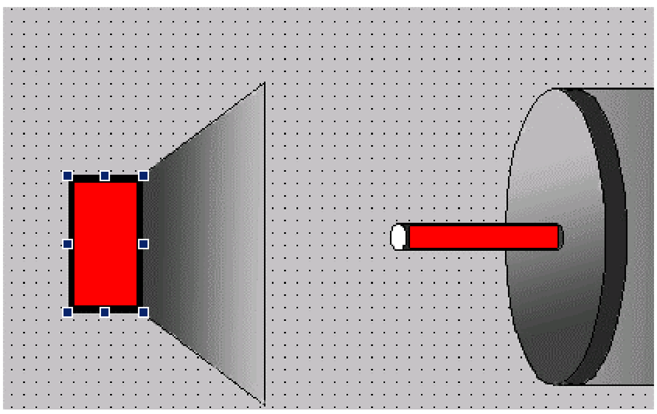
Situación de las etiquetas 'Motor Delay' y 'ms'

Por último debemos añadir los elementos que nos indiquen que la bocina y el motor están encendidos, para ello recurrimos a insertar 2 formas (shapes) diferentes:



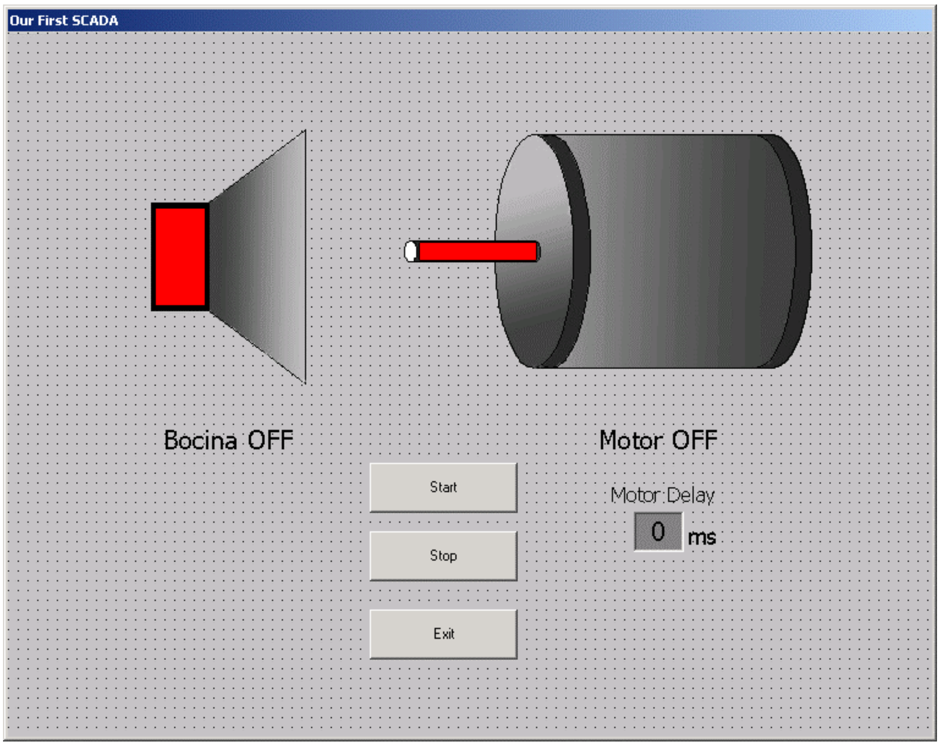
Insertar Forma (Shape)

Se insertan de igual manera que las imágenes y en sus propiedades podemos modificar el aspecto de las mismas. Insertamos dos shapes rectangulares y a las que le asignamos el color rojo.



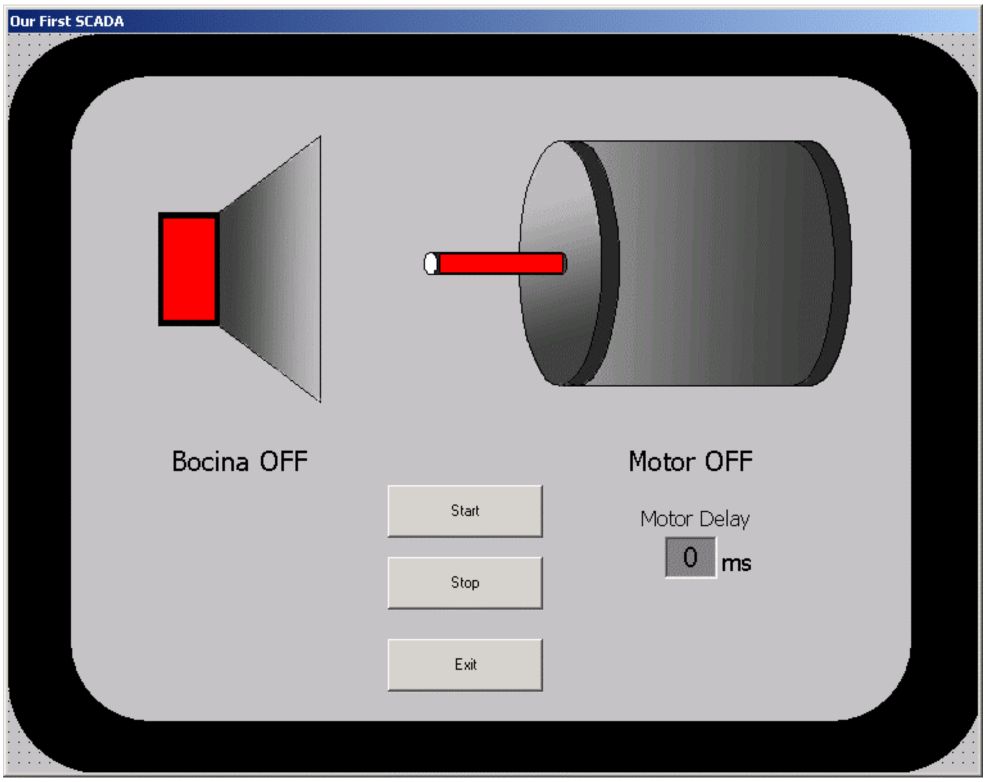
Se insertan de igual manera que las imágenes y en sus propiedades podemos modificar el aspecto de las mismas. Insertamos dos shapes rectangulares y a las que le asignamos el color rojo.

El aspecto que tendrá nuestro SCADA será el siguiente:



Aspecto Preeliminar del SCADA

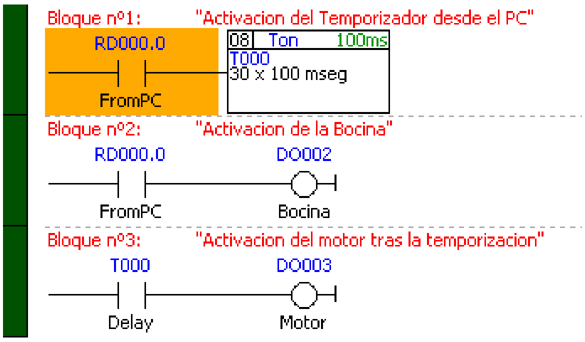
Podemos añadir formas (shapes) o imágenes adicionales sin ninguna función específica para enriquecer el aspecto del SCADA.



Aspecto Final del SCADA

Ahora debemos realizar los enlaces mediante la DLL para obtener la funcionalidad del SCADA.

Como podemos ver en el programa Ladder del MLCHIP1 tenemos:



Programa Ladder

Debemos monitorizar por tanto las siguientes variables: ‘DO002’, ‘DO003’ y el timer ‘T000’ así como controlar la variable ‘RD000.0’

Hacemos doble clic en una zona libre del formulario.

De esta manera accedemos al código del formulario , concretamente a la función Form_Load(). Está función se ejecuta cada vez que se carga el formulario, en nuestro caso esto es cada vez que arranquemos el SCADA.

```
Option Explicit

Private Sub Form_Load()

End Sub
```

Escribimos:

```
Option Explicit

Const ML_BIT As Integer = 0
Const ML_BYTE As Integer = 1
Const ML_WORD As Integer = 2
Const ML_DWORD As Integer = 3
Const ML_INT8U As Integer = 4
Const ML_INT8S As Integer = 5
Const ML_INT16U As Integer = 6
Const ML_INT16S As Integer = 7
Const ML_INT32U As Integer = 8
Const ML_INT32S As Integer = 9

Private Sub Form_Load()

    Call ML_OpenMonitor("COM1", "MLCHIP1", AddressOf myCallback)
    Call ML_InsertVariable("RD000.0", ML_BIT)
    Call ML_InsertVariable("DO002", ML_BIT)
    Call ML_InsertVariable("DO003", ML_BIT)
    Call ML_InsertVariable("T0", ML_INT8U)
    Call ML_InsertVariable("T0", ML_BIT)

    Call ML_StartMonitor
    Form1.Shape_bocina.Visible = False
    Form1.Shape_motor.Visible = False
    Form1.TextBoxDelay.Text = "0"
End Sub
```

Analicemos el código:

En el siguiente trozo de código que más tarde utilizaremos en el código fuente del SCADA, asociamos etiquetas de los diferentes tipo de datos / variables correspondientes a los tipos de variable que se puede encontrar en la programación de Microladder.

```
Const ML_BIT As Integer = 0
Const ML_BYTE As Integer = 1
Const ML_WORD As Integer = 2
Const ML_DWORD As Integer = 3
Const ML_INT8U As Integer = 4
Const ML_INT8S As Integer = 5
Const ML_INT16U As Integer = 6
Const ML_INT16S As Integer = 7
Const ML_INT32U As Integer = 8
Const ML_INT32S As Integer = 9
```

Lo primero de todo , es arrancar el programa de monitorización de la dll con:

```
Call ML_OpenMonitor("COM1", "MLCHIP1", AddressOf myCallback)
```

ML_OpenMonitor es una función de la DLL de Microladder, encargada de la inicialización de la monitorización.
Elegimos el Puerto serie y el chip conectado al PC que vamos a utilizar , en nuestro caso el puerto COM1 y el MLCHIP1. Al final de esta instrucción le decimos a la DLL que cuando el monitor reciba una trama de retorno desde el MLCHIP1 nuestra función 'myCallback' sea ejecutada. Dentro de ella haremos uso de las funciones 'ML_GetValue' y 'GetTimerValue' para que la DLL nos del valor de las variables monitorizadas y así poder dibujar por ejemplo, la forma (shape) de color rojo que indica la activación del motor.

```
Private Sub Form_Load()

    Call ML_OpenMonitor("COM1", "MLCHIP1", AddressOf myCallback)
    Call ML_InsertVariable("RD000.0", ML_BIT)
    Call ML_InsertVariable("DO002", ML_BIT)
    Call ML_InsertVariable("DO003", ML_BIT)
    Call ML_InsertVariable("T0", ML_INT8U)
    Call ML_InsertVariable("T0", ML_BIT)
```

ML_InsertVariable es otra función de la DLL de Microladder, cuyo objetivo es decirle al monitor que variables tiene que monitorizar en este caso DO002,DO003 y T000. Para T000 llamamos a la función dos veces, la primera para obtener el valor del timer, y la ultima para saber si el timer ha expirado.

```
    Call ML_StartMonitor
    Form1.Shape_bocina.Visible = False
    Form1.Shape_motor.Visible = False
    Form1.TextBoxDelay.Text = "0"
End Sub
```

ML_StarMonitor indica a la Dll que debe empezar a monitorizar y leer el ML-CHIP1

Las siguientes instrucciones se incluyen para inicializar algunos elementos del SCADA, en este caso cada vez que se carga el formulario las imágenes correspondientes a la bocina encendida y el motor encendido se inician como No-visibles y el texto que contiene los milisegundos restantes para la activación del motor se pone a "0".

```
Private Sub BotonStart_Click()
Call ML_PutBitValue("RD000.0", 1)
End Sub
```

```
Private Sub BotonStop_Click()
Call ML_PutBitValue("RD000.0", 0)
End Sub
```

```
Private Sub EXIT_Click()
    Call ML_StopMonitor
    End
End Sub
```

Estas instrucciones determinan que ordenes lanzan los botones cuando se hace clic en ellos:

Cuando se hace clic sobre el botón 'Start' se llama a la función ML_PutBitValue esta función se encarga de indicar al micro que debe poner una determinada variable tipo bit a "1" o a "0", se utiliza mucho para controlar conmutadores y pulsadores virtuales, su sintaxis es la siguiente:

ML_PutBitValue ("X", Y)

Donde "X" es la variable tipo bit que queremos controlar y que debe ser puesta entre comillas " ".
"Y" es el valor que queremos darle a la variable y debe ser "0" o "1"

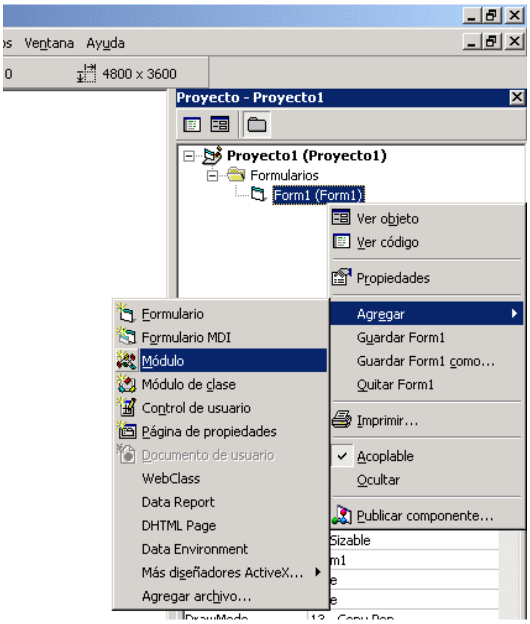
Por lo tanto si vemos el programa ladder, podemos comprobar que al hacer clic en el botón 'Start' ponemos a "1" el bit que activa el proceso de encendido del motor y la bocina

De la misma manera el botón 'Stop' llama a la misma función, solo que en este caso para poner el mismo bit a "0" y detener el motor y apagar la bocina

El botón "EXIT" llama a la función ML_StopMonitor encargada de detener la monitorización del micro.
Además ejecuta el comando end, que cierra el programa.

El Módulo:

Para poder hacer uso de las DLL de comunicación con ML-CHIP Debemos crear un módulo en el cual incluiremos la función de Callback, a la que puede dar el nombre que queramos, en nuestro caso 'myCallback'. Dentro de ella haremos uso de las funciones 'ML_GetValue' y 'GetTimerValue' para que la DLL nos del valor de las variables monitorizadas y así poder dibujar por ejemplo, la forma (shape) de color rojo que indica la activación del motor.



Crear Módulo

Una vez creado el módulo se puede escribir el código en de igual manera que lo hemos hecho en el formulario

```
Option Explicit
Dim nEstadoBocina As Integer
Dim nEstadoMotor As Integer
Dim nEstadoTimer As Integer


Function myCallback() As Integer

    If ML_TimeOutExceeded() = 1 Then
    Call MsgBox("Error en Comunicaciones")
    End If

    If nEstadoTimer <> ML_GetTimerValue("T0") Then
    nEstadoTimer = ML_GetTimerValue("T0")
    End If

    Form1.TextBoxDelay.Text = nEstadoTimer

    If nEstadoBocina <> ML_GetValue("DO002") Then
        nEstadoBocina = ML_GetValue("DO002")

        If nEstadoBocina = 1 Then
            Form1.bocina_txt.Caption = "Bocina ON"
            Form1.Shape_bocina.Visible = True
            Form1.bocina_txt.ForeColor = RGB(125, 0, 0)
        Else
            Form1.bocina_txt.Caption = "Bocina OFF"
            Form1.Shape_bocina.Visible = False
            Form1.bocina_txt.ForeColor = RGB(0, 0, 0)
        End If

    End If
```

```
    If nEstadoMotor <> ML_GetValue("DO003") Then
        nEstadoMotor = ML_GetValue("DO003")

        If nEstadoMotor = 1 Then
            Form1.motor_txt.Caption = "Motor ON"
            Form1.Shape_motor.Visible = True
            Form1.motor_txt.ForeColor = RGB(125, 0, 0)
        Else
            Form1.motor_txt.Caption = "Motor OFF"
            Form1.Shape_motor.Visible = False
            Form1.motor_txt.ForeColor = RGB(0, 0, 0)
        End If

    End If

    myCallback = 0

End Function
```

Código Fuente del Módulo

A continuación vamos a explicar que hace cada parte del módulo:

```
Option Explicit
Dim nEstadoBocina As Integer
Dim nEstadoMotor As Integer
Dim nEstadoTimer As Integer
```

En esta parte se definen tras el comando ‘Option Explicit’ las variables que vamos a utilizar en esta función.

Aunque no es necesario hacerlo, ‘Option Explicit’ nos obliga a que cualquier variable que utilicemos en e, programa este definida en esta zona, evitando que podamos utilizar variables que no existen u obsoletas por error. Hemos definido 3 variables como Integer, que utilizaremos a continuación.

```
Function myCallback() As Integer

    If ML_TimeOutExceeded() = 1 Then
    Call MsgBox("Error en Comunicaciones")
    End If
```

Llamamos a nuestra función ‘myCallback’

Este trozo de código usa el registro ML-TimeOutExceed el cual si no se ha podido establecer una comunicación por el puerto serie en tras un determinado tiempo, se pone a 1.
Si esto ocurre mostraremos mediante una ventana de text el mensaje “Error en Comunicaciones”

```
If nEstadoTimer <> ML_GetTimerValue("T0") Then
nEstadoTimer = ML_GetTimerValue("T0")
Form1.TextBoxDelay.Text = nEstadoTimer
End If
```

Comentemos que ocurre en el siguiente trozo de código :
ML-GetTimerValue es un comando que lee del micro el valor de la variable o del timer definido, en este caso se ha asociado al timer T0 (T000). Las dos primeras líneas de código tienes como misión leer del micro el valor del Timer “T0” y compararlo con el valor de “nEstadoTimer” si son diferentes, entonces el valor de timer se almacena en “nEstadoTimer de esta manera a la hora de refrescar gráficos del SCADA asociados a esta variable solo se hará cuando el valor de la misma cambie, evitando tener que refrescarlo en cada ciclo del programa. Con este método se evitan parpadeos innecesarios de los gráficos.

```
If nEstadoBocina <> ML_GetValue("DO002") Then
    nEstadoBocina = ML_GetValue("DO002")

    If nEstadoBocina = 1 Then
        Form1.bocina_txt.Caption = "Bocina ON"
        Form1.Shape_bocina.Visible = True
        Form1.bocina_txt.ForeColor = RGB(125, 0, 0)
    Else
        Form1.bocina_txt.Caption = "Bocina OFF"
        Form1.Shape_bocina.Visible = False
        Form1.bocina_txt.ForeColor = RGB(0, 0, 0)
    End If

End If
```

De la misma manera que hemos hecho con el timer, en las dos primeras líneas de esta parte de código, obtenemos y almacenamos el valor de DO002 (salida de la bocina según el programa ladder) a la variable nEstadoBocina.

El resto de código se encarga de realizar varias acciones para dos casos diferentes:

En el caso de que la bocina este a encendida (nEstadoBocina = 1) :

- El contenido de la etiqueta (label) “bocina_txt” pasará a ser “Bocina ON”
- La Forma de color Rojo “Shape_bocina” se hará visible
- El color de la etiqueta (label)de la bocina será el rojo .

En el caso de que la bocina este a apagada (nEstadoBocina = 0) :

- El contenido de la etiqueta (label) “bocina_txt” pasará a ser “Bocina OFF”
- La Forma de color Rojo “Shape_bocina” se hará invisible
- El color de la etiqueta (label) de la bocina será el negro .

Al igual que ocurría con la bocina haremos lo mismo con la salida del motor, atribuyéndole la variable “nEstadoMotor”

```
    If nEstadoMotor <> ML_GetValue("DO003") Then
        nEstadoMotor = ML_GetValue("DO003")

        If nEstadoMotor = 1 Then
            Form1.motor_txt.Caption = "Motor ON"
            Form1.Shape_motor.Visible = True
            Form1.motor_txt.ForeColor = RGB(125, 0, 0)
        Else
            Form1.motor_txt.Caption = "Motor OFF"
            Form1.Shape_motor.Visible = False
            Form1.motor_txt.ForeColor = RGB(0, 0, 0)
        End If

    End If

    myCallback = 0

End Function
```

En el caso de que el motor este funcionando (nEstadoMotor = 1) :

- El contenido de la etiqueta (label) “motor_txt” pasará a ser “Motor ON”
- La Forma de color Rojo “Shape_motor” se hará visible
- El color de la etiqueta (label) “motor_txt” del motor será el rojo .

En el caso de que la bocina este a apagada (nEstadoBocina = 0) :

- El contenido de la etiqueta (label) “motor_txt” pasará a ser “Motor OFF”
- La Forma de color Rojo “Shape_motor” se hará invisible
- El color de la etiqueta (label) “motor_txt” del motor será el negro.

Con esto termina el programa para el control y supervisón de una bocina y un motor.

Si pulsamos “run” (o tecla F5) el programa arranca y podemos comprobar su funcionamiento. No olvidemos que ninguna otra aplicación debe estar ocupando el puerto de comunicaciones, y que le programa debe de estar cargado en el chip y ejecutándose.

Resumen de los comandos de VB_SCADA_LADDER:

COMANDO	DEFINICIÓN
ML_OpenMonitor	Define el puerto, chip y función a utilizar.
ML_InsertVariable	Define las variables que utilizadas en el chip que queremos usar en el SCADA.
ML_StartMonitor	Comienza la monitorización del MLCHIP.
ML_StopMonitor	Detiene la monitorización del MLCHIP.
ML_PutBitValue	Modifica el valor de una variable tipo Bit.
ML_TimeOutExceeded	Registro que se pone a "1" cuando se produce un "timeOut" en las comunicaciones con ML CHIP
ML_GetValue	Lee del micro el valor de una determinada variable y el estado "0" o "1" de un timer.
ML_GetTimerValue	Lee el valor de un timer

Sintaxis de los comandos de **VB_SCADA_LADDER:**

EJEMPLOS DE SINTAXIS	
ML_OpenMonitor	ML_OpenMonitor ("COM1","MLCHIP1", AddressOf myCallback)
ML_InsertVariable	ML_InsertVariable ("RD002", INT16U)
ML_StartMonitor	ML_StartMonitor
ML_StopMonitor	ML_StopMonitor
ML_PutBitValue	ML_PutBitValue ("RD002.5", 1)
ML_TimeOutExceeded	ML_TimeOutExceeded ( )
ML_GetValue	ML_GetValue ("RD015") ; ML_GetValue ("T20")
ML_GetTimerValue	ML_GetTimerValue ("T0")